最短路

2019年3月11日　　14:54

-
- 任两点间最短路径

一. 图上任两点间最短路径

1. Floyd算法
   i. d[k][i][j]存储借助前k个结点从i到j的最短路长度
   ii. 初值：先**无穷大**，再有边权的**按边权**w赋值，最后自己到自己是**0**
   iii. 转移d[k][i][j]=min(d[k-1][i][j],d[k-1][i][k]+d[k-1][k][j])
   iv. ij循环不是阶段，只是为了方便统计的附加状态
   v. 空间优化：每一阶段k都之和上一轮k有关，可以通过k循环取消
   vi. 输入边权时w[i][j]可直接保存在d[i][j]作为新的初值
   vii.
```
for__(k,1,n)
    for__(i,1,n)
        for__(j,1,n)
            d[i][j]=min(d[i][j],d[i][k]+d[k][j]);
```
2. 应用：传递闭包
   i. 循环基本同上，转移方程改成`d[i][j] |= d[i][k] & d[k][j]`即可
3. 应用：无向图最小环（点数>=3的那种）
   i. 引：在开始第k轮循环前，d[i][j]存的是前k-1个点能使i到j的最短路，则d[i][j] + w[j][k] + w[k][i]即为包含i和j的最小环可能取值
   ii. 为了避免重复，循环应该形如
```
for__(k,1,n){
    for_(i,1,k)
        for_(j,1,i)
            ;//更新最小环
    for__(i,1,n)
        for__(j,1,n)
            ;//更新d
}
```
4. 例：健身房最佳选址
   i. 题目：二叉树形小区，每个结点都住了w个人，有父子关系的结点间路长为1，求所有人去健身房的总路长的最小值
   ii. 思路：有父子关系视作边权为1，用floyd求任两点间最短路，暴力求最小值即可，记得d和ans初值无穷大
```
for__(i,1,n){
    cin>>w[i]>>l>>r;
    if(l)
        d[i][l]=d[l][i]=1;
    if(r)
        d[r][i]=d[i][r]=1;
    d[i][i]=0;}
for__(k,1,n)
    for__(i,1,n)
        for__(j,1,n)
            d[i][j]=min(d[i][j],d[i][k]+d[k][j]);
```

```
for__(i,1,n){
	int t=0;
	for__(j,1,n)
		t+=d[i][j]*w[j];
	ans=min(ans,t);}
cout<<ans;
```

二. 树上任两点间最短路

1. 例题：给定全图最短路，求构造n条边，保证有解（CF100405A）
2. 引：易知n方条边求生成树，这n-1条边都是满足的
3. 转：第n条边需要特判一下，如果以上n-1条边能建出符合要求的全图最短路，那第n条边就是任意重边即可；如果要求全图最短路中有生成树上最短路更短的路，那那条边即为第n条边
4. 生成树上n方求最短路的函数见dfs

```
const int MN = 2005;

struct DSJ{  //并查集
	int rt[MN];  //他爹
	inline void init(int n){
		for(int i=1; i<=n; ++i) rt[i]=i;
	}
	inline int fd(int x){
		return x==rt[x] ? x : rt[x]=fd(rt[x]);
	}
	inline bool mg(int x,int y){     //返回是否合并过
		x=fd(x), y=fd(y);
		if(x==y) return 0;
		return rt[x]=y, 1;
	}
}dsj;
struct Edge{
	int x,y,v;
	Edge(int x,int y,int v):x(x),y(y),v(v){}
	bool operator<(const Edge&t){
		return v<t.v;
	}
};
int n, D[MN][MN], d[MN][MN];
bitset<MN>vst;
vector<Edge>v, ans;
vector<pair<int,int>> to[MN];
void krsk(){
	for(int i=1; i<=n; ++i) for(int j=1; j<=n; ++j) d[i][j]=0;
	for(int i=1; i<=n; ++i) for(int j=i+1; j<=n; ++j)
	v.emplace_back(i,j,D[i][j]);
	sort(v.begin(),v.end());
	int cnt=1;
	dsj.init(n);
	for(auto [x,y,z] : v){
		if(dsj.mg(x,y))
			++cnt,
			d[x][y]=d[y][x]=z,
			ans.emplace_back(x,y,z),
			to[x].emplace_back(y,z),
			to[y].emplace_back(x,z);
		if(cnt==n) break;   //剪枝
	}
}

void dfs(int x,int f,int dst,int anc){
	d[anc][x]=dst;
```

```
	for(auto [y,z]:to[x])
		if(y!=f)
			dfs(y,x,dst+z,anc);
}

int main(){
	while(~scanf("%d",&n)){
		for(int i=1; i<=n; ++i)
			for(int j=1; j<=n; ++j)
				scanf("%d",D[i]+j);
		krsk();

		for(int x=1; x<=n; ++x)
			dfs(x,0,0,x);

		bool isn=0;
		for(auto [x,y,z] : v){
			if(z<d[x][y]){
				isn=1;
				ans.emplace_back(x,y,z);
				break;
			}
		}
//for(auto [x,y,z]:v) printf("v %d %d %d\n",x,y,z);
//for(int x=1; x<=n; ++x) for(int y=1; y<=n; ++y) printf("%d%c",d[x]
[y]," \n"[y==n]);
		for(auto [x,y,z] : ans) printf("%d %d %d\n",x,y,z);
		if(!isn) printf("1 2 %d\n",d[1][2]);

		for(int i=1; i<=n; ++i) to[i].clear();
		ans.clear(); v.clear();
		puts("");
	}

	return 0;
}
```

三. 全图最短路

1. 例题：有向图最短路CF101498L
   i. 特判：全正权的图，输出最小边
   ii. 特判：有负环的图，输出-inf
   iii. 一般情况：建超级源点，求超级源点开始的最短路，即为全图最短路
   iv. 超级源点：除了从0点开始连0权边以外，在特判完全正权边的情况时，也可直接给每个点的dst初值都设为0，每个点都加入队列

```
const int MN = 7005<<1;
int n,m,tot,fst[2005];	//点数、边数、边号、链表首结点
ll dst[2005];//最短路
int to[MN],val[MN],nxt[MN];	//链表：终点、边权、下一边
inline void add(int x,int y,int v=0){
	to[++tot]=y;
	val[tot]=v;
	nxt[tot]=fst[x];
	fst[x]=tot;
}
int inq[MN];
queue<int> q;

bool spfa(int src){ //src为源点最短路，返回是否有环
	ms(dst,0x3f); dst[src]=0;
	ms(inq,0); inq[src]=1;
	while(q.size()) q.pop();
```

```
	q.push(src); //起点
	int cnt=0; q.push(cnt);
	if(src==0){ //超级源点
		for__(i,1,n) q.push(i), q.push(cnt);
		ms(dst,0);
	}
	while(!q.empty()){
		int x=q.front(); q.pop();
		cnt=q.front(); q.pop();
		if(cnt>n) return 1; //有负环
		inq[x]=0;	//x已不在队列中
		for(int i=fst[x]; i; i=nxt[i]){  //遍历x起点的各终点
			int y=to[i], z=val[i];
			if(dst[y]>dst[x]+z){
				dst[y]=dst[x]+z;
				if(!inq[y])  //y不在队列中
					q.push(y),
					q.push(cnt+1),
					inq[y]=1; }}}
	return 0;//没负环
}
inline void solve(){
	ms(fst,0); tot=0;
	scanf("%d%d",&n,&m);
	ll ans=1ll<<60;
	for_(i,0,m){
		int x,y,z;
		scanf("%d%d%d",&x,&y,&z);
		add(x,y,z);
		ans=min((ll)z,ans);
	}
	if(ans>=0) return printf("%lld\n",ans), void();
//	for__(i,1,n) add(0,i,0);
	if(spfa(0)) puts("-inf");
	else{
		for__(i,1,n) ans=min(ans,(ll)dst[i]);
		printf("%lld\n",ans);
	}
}
```